Instituto Tecnológico de Estudios Superiores de Monterrey

TE3001B.101

Fundamentación de Robótica

Challenge 3.

Gpo 101

Profesor:

Rigoberto Cerino Jiménez

Alumnos:

Daniela Berenice Hernández de Vicente	A01735346
Alejandro Armenta Arellano	A01734879
Dana Marian Rivera Oropeza	A00830027

Fecha: 06 de Marzo del 2023

Resumen

En este reporte se podrá observar una descripción de lo que es un PWM y cómo funciona, todo esto con tal de comprender más allá de simplemente resolver el challenge semanal impuesto por Manchester Robotics.
Aunado a estas descripciones y definiciones, se puede encontrar la solución paso por paso al challenge de esta tercera semana.

Objetivos

En este tercer challenge se espera que los estudiantes repasen los conceptos vistos en las sesiones pasadas.
Por lo cual la actividad consiste en lo siguiente:

- Crear varios nodos en ROS, los cuales deberán regular la velocidad de un Motor DC.
- El Motor deberá ser controlado por una computadora externa, un microcontrolador (Arduino) y el driver del motor.

Objetivos específicos:

- Nodo Motor:
    - El nodo motor debe ser ejecutado desde la placa Arduino.
    - Se debe suscribir al tópico “cmd_pwm”.
    - La salida del arduino debe ser la señal de dirección del motor driver junto con el mensaje de “PWM signal”.
        - El ciclo de trabajo y la dirección de PWM deben asignarse al intervalo $[-1, 1]$.
    - El controlador del motor debe enviar la potencia requerida al motor de CC..

Introducción

Semanalmente se nos presentará un reto por Manchester Robotics, los cuales serán resueltos con la ayuda de programas especializados como ROS y ciertos lenguajes de programación como Python.
De igual manera cada semana contaremos con una clase base con respecto a lo que se manejara durante el challenge semanal, así como una sesión de preguntas y respuestas donde se buscarán resolver dudas con respecto al mismo challenge de la semana.
Por otro lado también se nos proporcionarán algunas actividades las cuales hasta cierto punto serán un pequeño desglose del challenge semanal, claro está que contarán con una menor dificultad.
En esta tercera semana al ya contar con una noción clara de la arquitectura de trabajo de ROS, se nos introdujo a los siguientes temas:

- ROS Serial:
    - Rosserial fue creado con el propósito de estandarizar la comunicación entre las computadoras y el hardware robótico.
    - Rosserial es un protocolo para envolver mensajes serializados ROS estándar y multiplexar múltiples temas y servicios a través de un dispositivo de caracteres, como un puerto serie o un conector de red.
    - Permite que el hardware se comunique con el resto del sistema ROS.

    - Rosserial les permite utilizar temas, servicios y funciones de registro de ROS.
    - En resumen, permite crear Nodos ROS dentro de microcontroladores o hardware diferente, permitiendo la comunicación con diferentes sistemas dentro de la red ROS.
    - Rosserial tiene algunas limitaciones basadas en el microcontrolador a utilizar.
    - Uno de los más comunes es el tamaño máximo de un mensaje, número máximo de editores y suscriptores.
        - ATMEGA168 (Arduino UNO): búfer de entrada/salida de 150/150 bytes y 6 editores y 6 suscriptores.
        - ATMEGA328P (Arduino MEGA) Búfer de entrada/salida de 280/280 bytes y 25 editores y 25 suscriptores.
    - Uno de los usos más comunes de rosserial es usarse con un microcontrolador (como se describió anteriormente) para comunicarse con sensores, actuadores, etc.
    - A diferencia de una computadora que funciona con ROS, el sistema operativo dedicado de los microcontroladores permite al usuario tener más control sobre las funciones de sincronización requeridas para cierto hardware y algoritmos de control.
    - Arduino y ESP32 son algunos de los microcontroladores más comunes utilizados con rosserial.
    - Rosserial también se usa con otro hardware como Xbee, sensores y actuadores con Linux embebido, plataformas Mbed, etc.
    -
- ROS Comunicación Serial:
    - Como se indicó anteriormente, Arduino y ESP32 son algunos de los MCU más utilizados.
    - Ambos se pueden programar usando el IDE de Arduino.
    - Para todas las actividades y desafíos de esta sesión, se utilizará Arduino IDE para la programación.
    - Las actividades y los desafíos que se muestran en esta presentación se pueden realizar con MCR2 Hackerbard o Arduino Mega, junto con un motor de CC y un módulo controlador de motor (L298n) para el desafío.
    - Consulte los requisitos previos de esta sesión para obtener la lista completa de los componentes necesarios.
    - Arduino IDE:
        - Un IDE, o entorno de desarrollo integrado, ayuda a la productividad de los programadores mediante la combinación de actividades comunes de escritura de software en una sola aplicación: edición de código fuente, creación de ejecutables y depuración.
        - Arduino IDE es compatible con los lenguajes de programación C y C++.
        - Un boceto es un programa escrito con el IDE de Arduino.
        - Los bocetos se guardan en la computadora de desarrollo como archivos de texto con la extensión de archivo .ino.

- Sketch:
  - La sintaxis más simple para escribir un boceto consta de solo dos funciones:
  - setup (): esta función se llama una vez cuando se inicia un boceto después del encendido o reinicio. Se utiliza para inicializar variables, modos de pines de entrada y salida y otras bibliotecas necesarias en el boceto. Es análoga a la función main().
  - loop(): La función loop() se ejecuta repetidamente en el programa principal después de la función setup(). Controla la placa hasta que se apaga o se reinicia.

- ROS Estructura de Sketch:
  - Librerías de ROS en Arduino:
    - Rosserial proporciona un protocolo de comunicación ROS que funciona sobre Arduino/ESP32 UART.
    - Permite que Arduino/ESP32 se convierta en un nodo ROS que puede publicar y suscribirse directamente a mensajes ROS, publicar transformaciones TF y obtener la hora del sistema ROS.
    - Para instalar la biblioteca ROS para Arduino, siga los pasos de instalación en el ppt:
      - MCR2_ROS_ArduinoIDE_Configuración
  - Declaración de variables:
    - Para cada programa ROS Arduino, el encabezado ros.h debe incluirse antes que cualquier otra biblioteca o mensaje ROS.
    - Cree una instancia del identificador de nodo, que permite que nuestro programa cree editores y suscriptores. El controlador de nodo también se ocupa de las comunicaciones del puerto serie.
    - Cree una instancia de los editores, suscriptores que se utilizarán.
    - Declare todos los mensajes y variables de ROS que se van a utilizar.
    - Declarar/definir las funciones de devolución de llamada que se utilizarán con los suscriptores.
    - SetUp:
      - Inicialice su identificador de nodo ROS. Inicialización de nodos,
      - Anuncie cualquier tema que se esté publicando.
      - Suscríbete a los temas que se utilizan.
      - Inicializar variables, puertos, funciones, etc.
    - Loop:
      - Ejecuta el programa principal.
      - Bucle y repita acciones.
      - Manejar funciones de devolución de llamadas.
- Control de Motor DC:
  - Jerarquías de control:

        - Los sistemas de robótica, están destinados a realizar tareas complicadas en diferentes entornos. Dichas tareas se pueden realizar de forma autónoma, semiautónoma o por control remoto.
        - El control de este tipo de sistemas puede llegar a ser muy complejo dependiendo de la tarea a realizar.
        - El HCS divide el control en capas, dividiendo cada tarea compleja en subtareas (objetivos) que debe lograr una capa dedicada.
        - Este curso está dedicado a la capa de control de bajo nivel usando ROS.
    - Robots móviles:
        - Los robots móviles requieren diferentes sensores y actuadores para leer información del entorno e interactuar con ella.
        - Uno de los actuadores más comunes que se encuentran en la mayoría de los sistemas robóticos son los motores de CC.
        - Los motores de CC se encuentran en diferentes aplicaciones robóticas, desde robots móviles con ruedas hasta manipuladores robóticos y vehículos aéreos no tripulados.
        - Un ejemplo de robot móvil con ruedas es el Puzzlebot, que utiliza dos motores de corriente continua, uno para cada rueda.
        - Los motores de corriente continua son ampliamente estudiados en diferentes campos de la ciencia, desde sistemas electromecánicos hasta ingeniería de control.
    - Motor DC.
        - Un motor de corriente continua (CC) es un tipo de máquina eléctrica que convierte la energía eléctrica en energía mecánica.
        - Los motores de CC toman energía eléctrica a través de corriente continua y convierten esta energía en rotación mecánica.
        - Esto se hace mediante el uso de campos magnéticos generados por las corrientes eléctricas, impulsando el movimiento de un rotor fijo dentro del eje de salida.
        - El par y la velocidad de salida dependen tanto de la entrada eléctrica como del diseño del motor.
        - En robótica, los controladores se utilizan para regular la velocidad de rotación, la posición angular o el par requerido por la aplicación.
        - En robótica esto se llama control de bajo nivel.
        - Para el caso de un robot móvil con ruedas es una práctica común implementar un control PID para regular la velocidad angular de los motores DC.
        - La regulación de la velocidad angular o posición de un motor, requiere de diferentes etapas.
            - Etapa del controlador
            - Etapa de potencia (controlador)
            - Planta
            - Etapa del sensor

- Controlador del Motor:
  - El puente H es un circuito electrónico que cambia la polaridad de un voltaje aplicado a una carga.
  - Funcionan utilizando una combinación de componentes de conmutación (interruptores mecánicos, transistores, etc.), como se muestra en el diagrama, para cambiar la polaridad de la carga.
  - Los controladores H-Bridge son algunos de los controladores de motor más comunes que se utilizan en los motores de CC de control para funcionar hacia adelante o hacia atrás.
  - Otra capacidad del controlador de motor es regular la velocidad angular del motor.
  - Hay muchas formas de obtener este resultado, una de las más comunes es enviar una PWM (señal modulada por ancho de pulso) al pin de habilitación del H-Bridge.
  - PWM (modulación de ancho de pulso): es una técnica utilizada en ingeniería para controlar la potencia promedio entregada por una señal eléctrica, dividiéndola en partes discretas.
  - En la práctica, esto se logra encendiendo y apagando rápidamente el interruptor entre la carga y la fuente (interruptor de habilitación).
  - Dado que el motor se puede modelar como un sistema de segundo orden, al aplicar un voltaje PWM como entrada, es posible observar el comportamiento de salida como se muestra en la figura.
  - Este comportamiento se puede utilizar para controlar la potencia entregada al motor y, por lo tanto, controlar la velocidad angular del motor.
- MCU:
  - Un microcontrolador es un circuito integrado compacto. Están hechos para realizar una operación específica en un sistema embebido.
  - En robótica suelen estar a cargo del control de bajo nivel del robot, como motores, sensores o actuadores que requieren un controlador dedicado y rápido para funcionar.
  - Un microcontrolador típico incluye un procesador, memoria y periféricos de entrada/salida (E/S) en un solo chip.}

Solución del problema

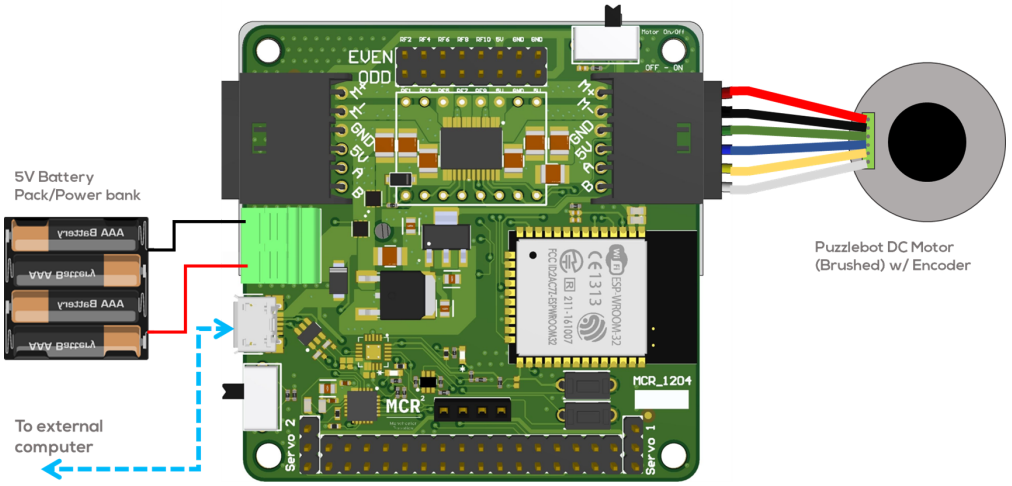


Diagrama de conexión 1

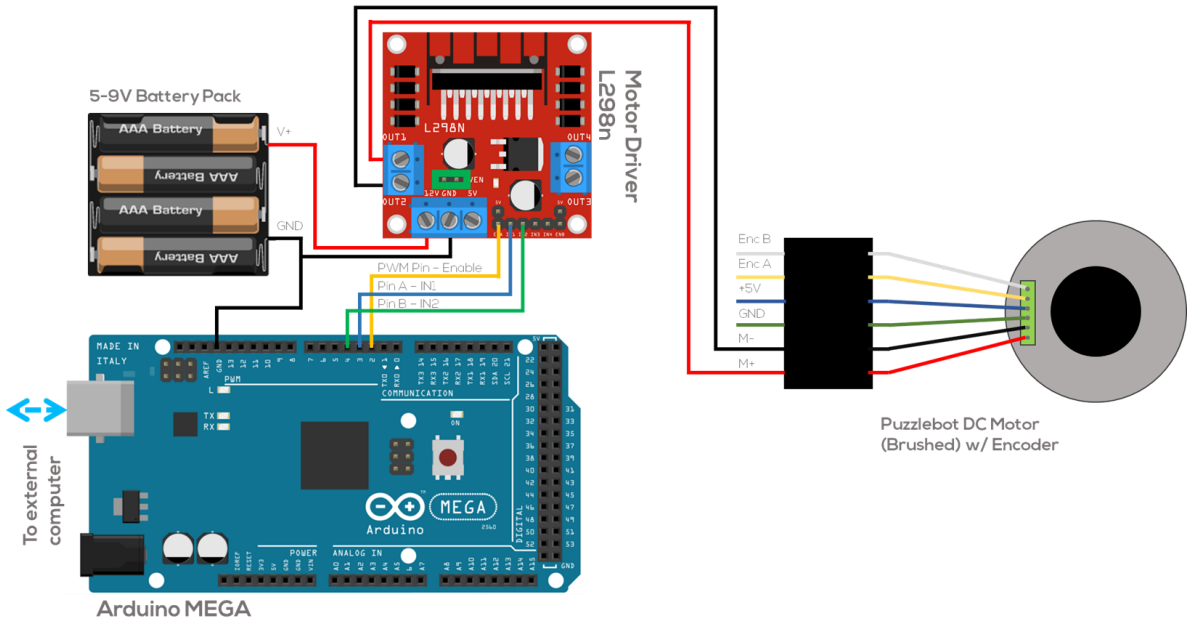


Diagrama de conexión 2

Por otro lado, los comandos principales para el código han sido vistos en los challenges anteriores, por lo cual en este los hemos omitido.
Aunado a esto, describimos los códigos implementados en la solución del challenge semanal.

Nodo Signal_generator:

```python
#!/usr/bin/env python
import rospy
import numpy as np
import sys
from std_msgs.msg import Float32
import math

t = rospy.get_param("t",1)



if __name__=='__main__':
    rospy.init_node("signal_generator")
    pub_signal=rospy.Publisher("cmd_pwm", Float32, queue_size=10)
    pub_time=rospy.Publisher("time", Float32, queue_size=10)
    rate=rospy.Rate(5)
    t=0

    while not rospy.is_shutdown():
        t+= 0.1
        y=t*(np.sin(t))
        y=(y+20)*75

        print_info = "%3f | %3f" %(y,t)
        rospy.loginfo(print_info)
        pub_signal.publish(y)
        pub_time.publish(t)
        rate.sleep()

```

En este nodo lo que se hace es generar la señal con respecto al motor DC y publicarla mediante el tópico “cmd_pwm” donde se imprime en el rosserial como podremos observar en la sección de resultados.

Nodo Serial_node:

```
1
2   #include <ros.h>
3   #include <std_msgs/Float32.h>
4
5   ros::NodeHandle  nh;
6
7   void messageCb( const std_msgs::Float32& msg){
8     Serial.println(msg.data);
9
10  }
11
12  ros::Subscriber<std_msgs::Float32> sub("signal", &messageCb );
13
14  void setup()
15  {
16    nh.initNode();
17    nh.subscribe(sub);
18    Serial.begin(9600);
19  }
20
21  void loop()
22  {
23    nh.spinOnce();
24    delay(1);
25
```

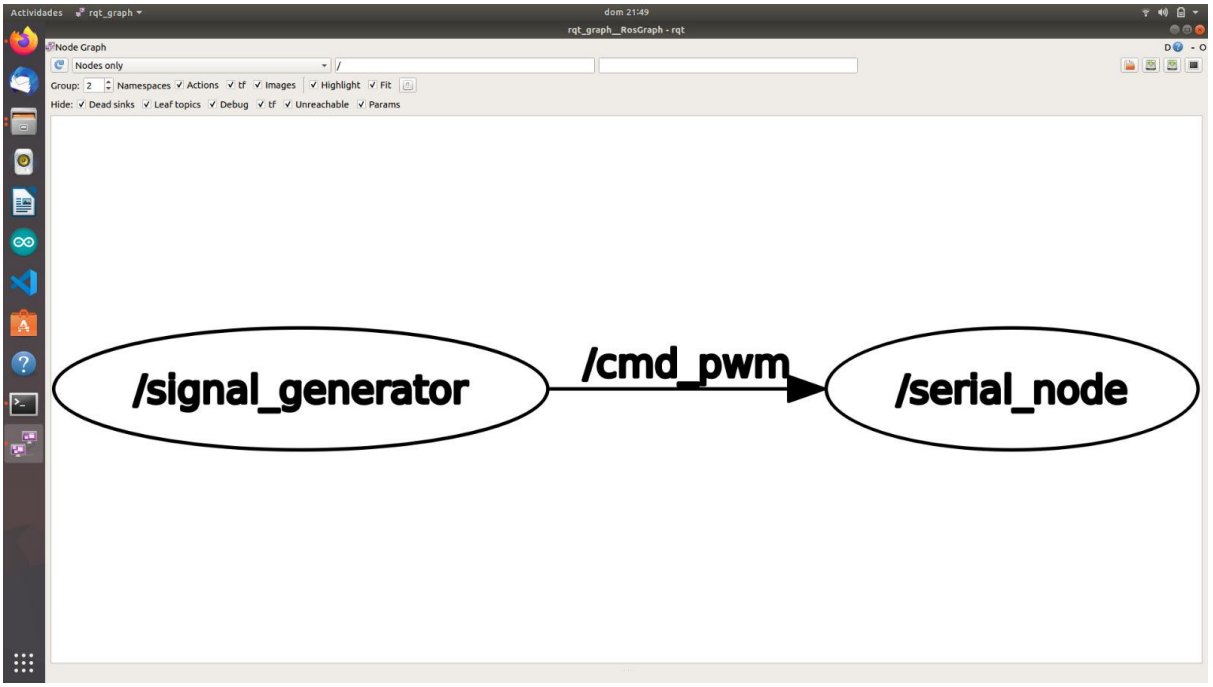
En este código se observa la manera en la cual conseguimos que ROS y la placa de arduino interactúen, cabe destacar que el código se genera mediante el IDE, donde lo que se hace es leer los datos y filtrarlos.

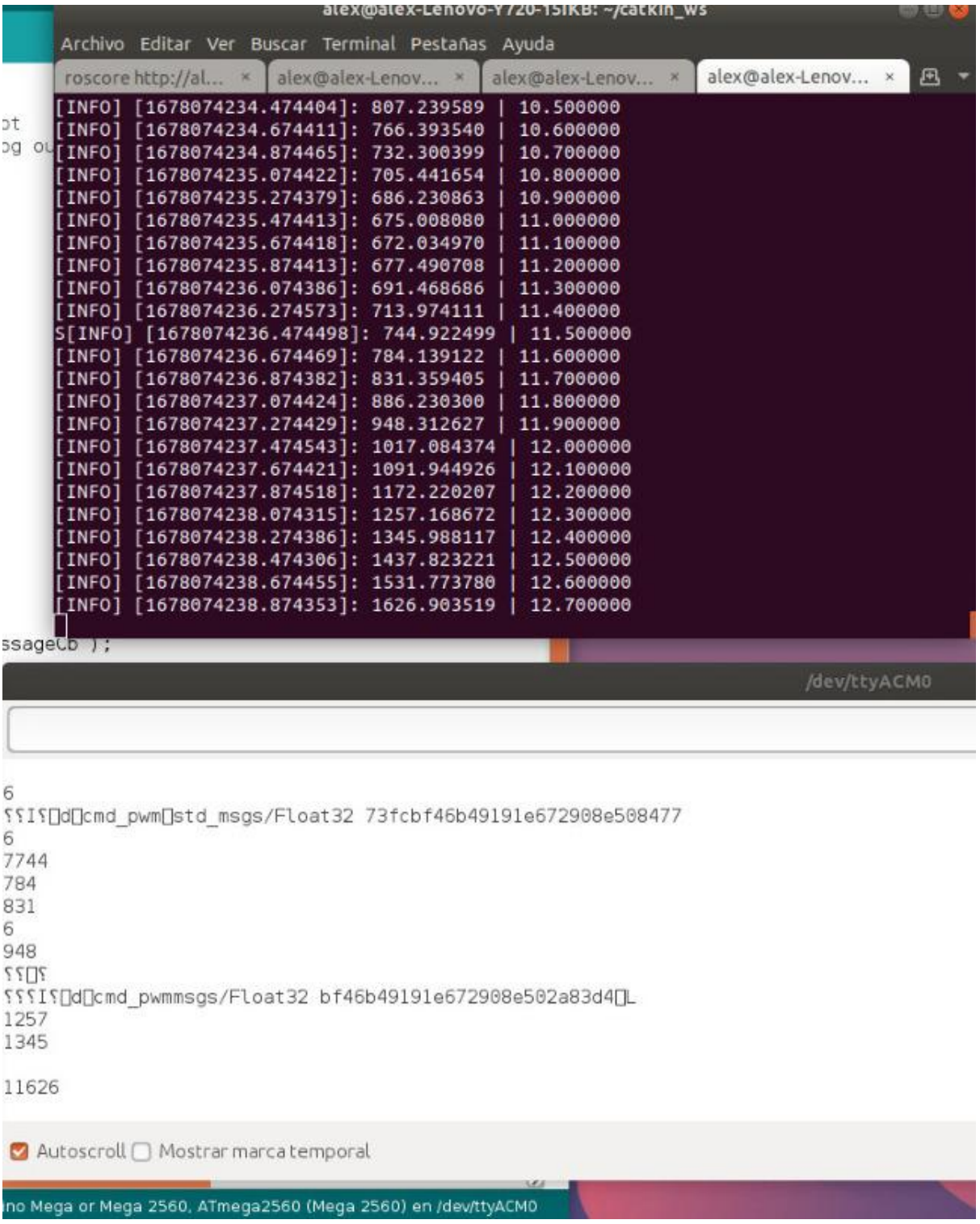
En este código se crea un control en la entrada del nodo sistema. De igual manera el nodo pública en el tópico “motor_input” y se suscribe a los tópicos “motor_output” y “set_point”. La salida del controller “motor_input” está delimitado entre el intervalo -1 a 1 i.e., $u(k) \in [-1,1]$ y se mandan los mensajes personalizados por nosotros.

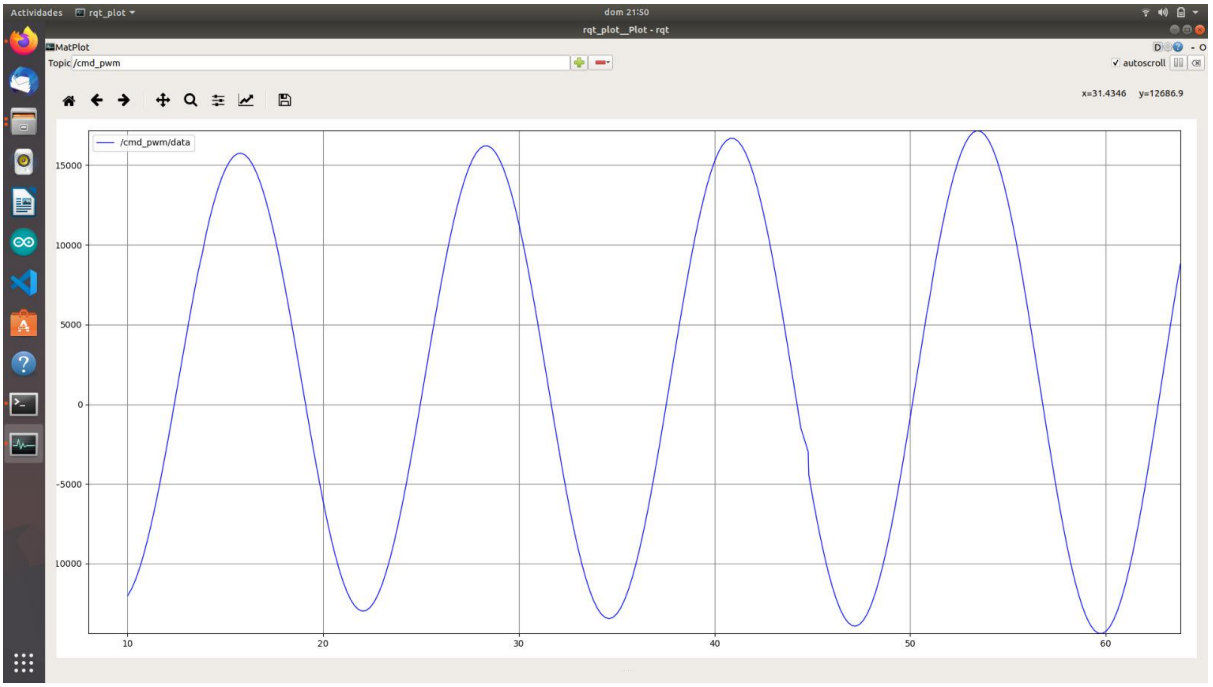
Resultados

En este apartado se muestra la relación que existe entre los canales de comunicación y entre los nodos para la transmisión de la señal de entrada y salida, y su procesamiento.
Todo esto gracias a que el nodo signal_generator manda la información a través del tópico “cmd_pwm” el cual llega al segundo nodo “serial_node” donde la información se pública a tráves del Rosserial, donde usamos la placa y el IDE arduino, para las conexiones y lecturas.

De igual manera anexamos una captura tanto de la senoidal que nos arroja el código como de los diferentes valores, destacando que el serial de arduino recibe datos “basura” ya que el serial se encuentra ocupado por el Rosserial.


Actividades
rqt_graph
dom 21:49
rqt_graph__RosGraph - rqt
Node Graph
Nodes only
/
Group: 2
Namespaces
Actions
tf
Images
Highlight
Fit
Hide:
Dead sinks
Leaf topics
Debug
tf
Unreachable
Params
/signal_generator
/cmd_pwm
/serial_node


alex@alex-Lenovo-Y720-15IKB: ~/catkin_ws
Archivo Editar Ver Buscar Terminal Pestañas Ayuda
roscore http://al...
alex@alex-Lenov...
alex@alex-Lenov...
alex@alex-Lenov...
[INFO] [1678074234.474404]: 807.239589 | 10.500000
[INFO] [1678074234.674411]: 766.393540 | 10.600000
[INFO] [1678074234.874465]: 732.300399 | 10.700000
[INFO] [1678074235.074422]: 705.441654 | 10.800000
[INFO] [1678074235.274379]: 686.230863 | 10.900000
[INFO] [1678074235.474413]: 675.008080 | 11.000000
[INFO] [1678074235.674418]: 672.034970 | 11.100000
[INFO] [1678074235.874413]: 677.490708 | 11.200000
[INFO] [1678074236.074386]: 691.468686 | 11.300000
[INFO] [1678074236.274573]: 713.974111 | 11.400000
S[INFO] [1678074236.474498]: 744.922499 | 11.500000
[INFO] [1678074236.674469]: 784.139122 | 11.600000
[INFO] [1678074236.874382]: 831.359405 | 11.700000
[INFO] [1678074237.074424]: 886.230300 | 11.800000
[INFO] [1678074237.274429]: 948.312627 | 11.900000
[INFO] [1678074237.474543]: 1017.084374 | 12.000000
[INFO] [1678074237.674421]: 1091.944926 | 12.100000
[INFO] [1678074237.874518]: 1172.220207 | 12.200000
[INFO] [1678074238.074315]: 1257.168672 | 12.300000
[INFO] [1678074238.274386]: 1345.988117 | 12.400000
[INFO] [1678074238.474306]: 1437.823221 | 12.500000
[INFO] [1678074238.674455]: 1531.773780 | 12.600000
[INFO] [1678074238.874353]: 1626.903519 | 12.700000
ssageCb);
/dev/ttyACM0
6
ʕʕIʕ□d□cmd_pwm□std_msgs/Float32 73fcbf46b49191e672908e508477
6
7744
784
831
6
948
ʕʕ□ʕ
ʕʕʕIʕ□d□cmd_pwmmsgs/Float32 bf46b49191e672908e502a83d4□L
1257
1345
11626
Autoscroll
Mostrar marca temporal
no Mega or Mega 2560, ATmega2560 (Mega 2560) en /dev/ttyACM0



Conclusiones

Aún cuando en algunos momentos parecía inalcanzable, se consiguió realizar con éxito cada uno de los objetivos antes mencionados.

Se logró la lectura de los datos, haciendo que en determinado rango el motor cambie su sentido de giro..

Se logró obtener los resultados esperados por MCR2.